IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In re patent application of:
Fontoura et al.

Serial No.: 10/723,391

Filed: November 25, 2003

Group Art Unit: 2168

Examiner: Wong, Joseph D.

Atty. Docket No.: ARC9920030080US1

For: USING INTRA-DOCUMENT INDICES TO IMPROVE XQUERY PROCESSING OVER XML STREAMS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' APPEAL BRIEF

Sirs:

Appellant respectfully appeals the final rejection of claims 1, 13, 25, and 37-53, in the Office Action dated January 25, 2008. A Notice of Appeal was timely filed on April 17, 2008.

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, Armonk, New York, assignee of 100% interest of the above-referenced patent application.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1, 13, 25, and 37-53, all the claims pending in the application and set forth fully in the attached appendix (Section IX), are under appeal. Claims 1-37 were originally filed in the application. A non-final Office Action was issued on September 7, 2007 rejecting claims 1, 13, 25, and 37. The Appellants filed an Amendent under 37 C.F.R. §1.111 on December 6, 2008 amending claims 1, 13, 25 and 37, and adding claims 38-53. A final Office Action was issued on January 25, 2008 rejecting claims 1, 13, 25, and 37-53. The Appellants filed a Request for Reconsideration under 37 C.F.R. §1.116 on March 24, 2008. An Advisory Action was issued on April 11, 2008 indicating that the Request for Reconsideration filed under 37 C.F.R. §1.116 on March 24, 2008 would be entered. The Appellants filed a Notice of Appeal timely on April 17, 2008.

Claims 1, 13, 25, and 37-53 stand rejected under 35 U.S.C. §102(e) as anticipated by U.S. Patent Application Publication No. 2003/0159112 to Fry hereinafter referred to as "Fry".

Appellants respectfully traverse these rejections based on the following discussion.

IV. STATEMENT OF AMENDMENTS

An final Office Action dated January 25, 2008 stated all the pending claims 1, 13, 25, and 37-53 were rejected. The claims shown in the appendix (Section IX) are shown in their amended form as of the March 24, 2008 Request for Reconsideration.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The Summary of the claimed subject matter is shown by the following annotated, independent claims, wherein paragraph numbers are designated as, for example, [0001], page and line numbers correspond to the designated lines of the paragraph, and Figures are followed by an element number, where appropriate:

Independent Claim 1

A method for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document ([0022], p. 8, l. 14-16; [0023], p. 9, l. 4-7) said method comprising:

producing, by said streaming API for a mark-up language data stream, an ordered index ([0020], p. 7, l. 7-9 and 14-15; Fig. 1) of all textual elements corresponding to their order in said mark-up language data stream ([0022], p. 8, l. 16-17; [0027], p. 10, l. 17-19), said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements ([0029], p. 13, l. 1-2; [0020], l. 14-15; Fig.1)

scanning, by a processor ([0021], p. 8, l. 3-4; Fig. 2(a);), all tag identifiers ([0029], p. 13, l. 1-2) of said ordered index to determine if there exists a match between a query and any of said tag identifiers ([0028], p. 10, l. 20 to p. 11, l.30; [0030]);

parsing a matched textual element ([0021], p. 8, l. 4-7; Fig. 2(a)), if a tag identifier ([0029], p. 13, l. 1-2), corresponding to said matched textual element, matches said query ([0030], p.13, l. 7-9); and

skipping an unmatched textual element for parsing ([0020], p. 8, l. 5-7), if a tag

identifier ([0029], p. 13, l. 1-2), corresponding to said unmatched textual element, does not match said query ([0030, p. 13, l. 9-16).

Independent Claim 13

A system for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document (Fig. 2(b)), said system comprising:

an ordered index (Fig. 2(a)) of all textual elements corresponding to their order in said mark-up language data stream ([0022], p. 8, l. 16-17; [0027], p. 10, l. 17-19), said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming API for a mark-up language data stream ([0029], p. 13, l. 1-2; [0020], p. 7, l. 14-15);

a processor (Fig. 2(a); [0021, p. 8, l. 3-4) adapted to scan all tag identifiers ([0029], p. 13, l. 1-2) of said ordered index to determine if there exists a match between a query and any of said tag identifiers ([0028], p. 10, l. 20 to p. 11, l. 30; [0030]); and

a parser adapted to parse a matched textual element (Fig. 2(a); [0021], p. 8, l. 4-7); if a tag identifier ([0029], p. 13, l. 1-2) corresponding to said matched textual element, matches said query ([0030], p. 13, l. 7-9), and to skip an unmatched textual element for parsing, if a tag identifier ([0029], p. 13, l. 1-2), corresponding to said unmatched textual element, does not match said query ([0030], p. 13, l. 9-16).

Independent Claim 25

A program storage device readable by computer comprising a program of instructions executable by said computer to perform a method for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document (Fig. 3, 11 and 13; [0024], p. 9, l. 14), said method comprising:

producing, by said streaming API for a mark-up language data stream, an ordered index ([0020], p. 7, l. 7-9 and 14-15; Fig. 1) of all textual elements corresponding to their

order in said mark-up language data stream ([0022], p. 8, l. 16-17; [0027], p. 10, l. 17-19), said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements ([0029], p. 13, l. 1-2; [0020], l. 14-15; Fig.1)

scanning, by a processor ([0021], p. 8, l. 3-4; Fig. 2(a);), all tag identifiers ([0029], p. 13, l. 1-2) of said ordered index to determine if there exists a match between a query and any of said tag identifiers ([0028], p. 10, l. 20 to p. 11, l.30; [0030]);

parsing a matched textual element ([0021], p. 8, l. 4-7; Fig. 2(a)), if a tag identifier ([0029], p. 13, l. 1-2), corresponding to said matched textual element, matches said query ([0030], p.13, l. 7-9); and

skipping an unmatched textual element for parsing ([0020], p. 8, l. 5-7), if a tag identifier ([0029], p. 13, l. 1-2), corresponding to said unmatched textual element, does not match said query ([0030, p. 13, l. 9-16).

Independent Claim 37

A system for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document (Fig. 2(b)), said system comprising:

an ordered index (Fig. 2(a)) of all textual elements corresponding to their order in said mark-up language data stream ([0022], p. 8, l. 16-17; [0027], p. 10, l. 17-19), said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming API for a mark-up language data stream ([0029], p. 13, l. 1-2; [0020], p. 7, l. 14-15);

a processor (Fig. 2(a); [0021, p. 8, l. 3-4) adapted to scan all tag identifiers ([0029], p. 13, l. 1-2) of said ordered index to determine if there exists a match between a query and any of said tag identifiers ([0028], p. 10, l. 20 to p. 11, l. 30; [0030]); and

a parser adapted to skip an unmatched textual element, if a tag identifier ([0029], p. 13, l. 1-2), corresponding to said unmatched textual element, does not match said query ([0030], p. 13, l. 9-16).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented for review by the Board of Patents Appeals and Interferences are whether claims 1, 13, 25, and 37-53 are unpatentable under 35 U.S.C. §102(e) as anticipated by Fry.

VII. ARGUMENT

A. The Rejection of Claims 1, 13, 25, and 37-53 under 35 U.S.C. §102(e)

1. The Position in the Final Office Action

Regarding claim 1, Fry teaches a method for (interpreted to be an intended use) query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document (see Abstract, cover Fig., wherein the textual limitation is met text shown in Fig. 3), said method comprising: producing5 by said streaming API for a mark-up language data stream (Abstract, wherein a mark-up language data stream is interpreted to include an "XML stream"), an ordered index of all textual elements corresponding to their order in said mark-up language data stream (Fig. 3, wherein the elements are shown in an order), said ordered index comprising tag identifiers and end positions corresponding to each of said 7 all textual elements ([30, 35], wherein navigation to the end of a document is discussed, also see claim 1); scanning by a processor (Fig. 1, item 106 meets the limitation), all tag identifiers of said ordered index to determine if there exists a 7, match between a query and any of said tag identifiers ([31], wherein the ordered index is stepped until the element to be extracted is processed by the base parser); parsing a matched textual element ([31], supra) 1 if interpreted to be 7 conditional) a tag identifier ([31], wherein the tag identifier is interpreted to be an "element tag"), corresponding to said matched textual element, matches said query ([311, wherein iterative stepping conditioned on a match meets the limitation); and skipping an unmatched textual element for parsing ([28-30] wherein the skip function allows the programmer to "skip ahead to specific sections.. .or get subsections"), if (interpreted to be conditional) a tag identifier, corresponding to said unmatched textual element, does not match said query. ([28-30], wherein "Element type two ends when another end tag is reached in the document", also see Fig. 3)

Regarding claim 13, Fry teaches a system for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document (see Abstract, cover Fig., wherein the textual limitation is met text shown in Fig. 3), said system comprising: an ordered index of all textual elements corresponding to their order in said mark-up language data stream (Abstract, wherein a

mark-up language data stream is interpreted to include an "XML stream"), said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming API for a mark-up language data stream (Fig. 3, wherein the elements are shown in an order) ; a processor (Fig. 1, item 106 meets the limitation) adapted to scan all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers ([30, 35], wherein movement to the end of a document is discussed, also see claim 1) ; and a parser adapted to parse a matched textual element ([311, supra), if a tag identifier corresponding to said matched textual element ([311, wherein iterative stepping conditioned on a match meets the limitation), matches said query, and to skip an unmatched textual element for parsing ([28-30), wherein the skip function allows the programmer to "skip ahead to specific sections.. .or get subsections"), if (interpreted to be conditional) a tag identifier, corresponding to said unmatched textual element, does not match said query. ([28-30], wherein "Element type two ends when another end tag is reached in the document", also see Fig. 3)

Regarding claim 25, Fry teaches a program storage device readable by computer comprising a program of instructions executable by said computer to perform a method for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document (see Abstract, cover Fig. , wherein the textual limitation is met text shown in Fig. 3), said method comprising: producing, in said streaming API for a mark-up language data stream (Abstract, wherein a mark-up language data stream is interpreted to include an "XML stream") , an ordered index of all textual elements corresponding to their order in said mark-up language data stream (Fig. 3, wherein the elements are shown in an order), said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements (Fig. 3, wherein the elements are shown in an order); scanning by a processor (Fig. 1, item 106 meets the limitation), all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; parsing a matched textual element, if a tag identifier (interpreted to be conditional), corresponding to said matched textual element ([31], supra), matches said query; and skipping an unmatched textual element for parsing ([28-30], wherein the skip function allows the programmer to "skip ahead to specific sections. . .or get subsections"), if a tag identifier (interpreted to be conditional), corresponding to said unmatched textual element, does not match said query. ([28-30], wherein "Element type two ends when another end tag is reached in the document", also see Fig. 3).

Regarding claim 37, Fry teaches a system for a query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document (see Abstract, cover Fig., wherein the textual limitation is met text shown in Fig. 3) , said system comprising: an ordered index of all textual elements corresponding to their order in said mark-up language data stream (Abstract, wherein a

mark-up language data stream is interpreted to include an "XML stream"; Fig. 3, wherein the elements are shown in an order), said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming API for a mark-up language data stream (see Abstract, Fig. 3); a processor (Fig. I, item 106 meets the limitation) adapted to scan all tag- identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers ([31], wherein iterative stepping conditioned on a match meets the limitation); and a parser adapted to skip an unmatched textual element, if (interpreted to be conditional) a tag identifier , corresponding to said unmatched textual element [31] , supra) , does not match said query. ([28-30] , wherein "Element type two ends when another end tag is reached in the document", also see Fig. 3).

Regarding claim 38, Fry teaches the method, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprised a tag identifier and an end position. ([28-31], Figs. 2-3).

Regarding claim 39, Fry teaches the method, all the limitations of which are incorporated herein by reference, further comprising storing a parsed matched textual element in a buffer. ([28-31], Figs. 2-3).

Regarding claim 40, Fry teaches the method, all the limitations of which are incorporated herein by reference, wherein upon said skipping of said unmatched textual element for parsing, said method further comprises offsetting said parser based upon an end position stored in said ordered index and corresponding to said unmatched textual element. ([28-31], Figs. 1-3, limitation is interpreted to be a negative).

Regarding claim 41, Fry teaches the method, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) (interpreted to be an optional alternative) or Extensible Markup Language (XML). (Abstract).

Regarding claim 42, Fry teaches the system, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprising a tag identifier and an end position. ([28-31], Figs. 2-3).

Regarding claim 43, Fry teaches the system, all the limitations of which are incorporated herein by reference, further comprising a buffer, operatively connected to said parser, in which a parsed matched textual element is stored. ([28-311, Figs. 1-3)

Regarding claim 44, Fry teaches the system, all the limitations of which are incorporated herein by reference, wherein said parser is offset based upon an end position stored in said ordered index that corresponds to said unmatched textual element. ([28-311, Figs. 2-3).

Regarding claim 45, Fry teaches the system, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) (interpreted to be an optional alternative) or Extensible Markup Language (XML). (Abstract).

Regarding claim 46, Fry teaches the program storage device, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual subelements, and each of said textual sub-elements comprises a tag identifier and an end position. ([28-311, Figs. 1-3).

Regarding claim 47, Fry teaches the program storage device, all the limitations of which are incorporated herein by reference, further comprising storing a parsed matched textual element in a buffer. ([28-31], Figs. 1-3).

Regarding claim 48, Fry teaches the program storage device, all the limitations of which are incorporated herein by reference, wherein upon said skipping if said unmatched textual element for parsing, said method further comprises offsetting said parser based upon an end position stored in said ordered index and corresponding to said unmatched textual element. ([28-31], Figs. 1-3, limitation is interpreted to be a negative).

Regarding claim 49, Fry teaches the program storage device, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) (interpreted to be an optional alternative) or Extensible Markup Language (XML). (Abstract).

Regarding claim 50, Fry teaches the system, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprising a tag identifier and an end position. ([28-311, Figs. 1-3).

Regarding claim 51, Fry teaches the system, all the limitations of which are incorporated herein by reference, further comprising a buffer, operatively connected to said parser, in which a parsed matched textual element is stored. ([28-31], Figs. 1-3)

Regarding claim 52, Fry teaches the system, all the limitations of which are incorporated herein by reference, wherein said parser is offset based upon an end position stored in said ordered index that corresponds to said unmatched textual element. ([28-311, Figs. 1-3, limitation is interpreted to be a negative).

Regarding claim 53, Fry teaches the system, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) (interpreted to be an optional alternative) or Extensible Markup Language (XML). (Abstract).

2. The Prior Art Reference

Fry discloses that "[b]road XML support is obtained through use of a set of streaming parser APIs. An application or client needing access to an XML document can contact an XML parser, XML processor, or XML reader in order to gain access to the document. The XML parser selects and instantiates a streaming parser API that is appropriate for the XML document and the client or application. Streaming parser APIs

include raw streaming parser APIs, non-validating streaming parser APIs, and validating streaming parser APIs. The XML parser can then provide a variety of types of access to the application or client that does not require the entire document to be read into memory, including providing an XML stream, pulling XML information, and skipping unwanted XML from the document." (Abstract, which is cited by the Final Action).

Fry also discloses that "a programmer can ask for the next event [i.e., next SAX event], or pull the next event, rather than handling the event in a callback. This gives the programmer more procedural control over the processing of the XML document. A streaming parser also allows the programmer to stop processing the document, skip ahead to specific sections of the document, and/or get subsections of the document as mini DOM trees." (Paragraph [0028], which is cited by the Final Action).

Fry further discloses that "Fig. 3 illustrates an event stream, with methods being used to manipulate the current position in the stream. The column on the left represents the XML document, the column on the right represents the Java code, and the column in the middle represents the event stream. In the Figure, the Java method startDocument( ) is shown to correspond to the Processing Instruction 300 of the event stream, which looks to the header of the XML document. The Java method startElement( ) is called and passed with the event "doc", which corresponds to the StartElementEvent:doc event 302 or the <doc> tag in the XML element. At the first element in the body of the XML document, given here as type "one", a startElement( ) method is again called, but with the element property corresponding to StartElementEvent:element 304 event in the event stream. In the XML document, a,/element> end tag signifies the end of the element, corresponding to an EndElementEvent 308in the event stream." (Paragraph [0029], which is cited by the Final Action).

Fry yet further discloses that "the parser would then reach element type "two" in the XML document, corresponding to another StartElementEvent:element 310 in the event stream. This would generate a substream in the Java environment to handle the second element type. Values 312, 314, 316 of element type "two" are placed onto the event stream and correspond to the Java substream. Element type "two" ends when

another end tag is reached in the document, corresponding to an EndElementEvent 318 in the event stream, with another EndElementEvent 320 corresponding to the end of the document tag </doc>." (Paragraph [0030], which is cited by the Final Action).

Fry yet further discloses that "[i]n a method for utilizing such an event stream, the name of the element to be extracted from the XML document is passed to an iterative method built on top of a base parser, such as a SAX API or DOM parser. An element tag of the XML document is located and the element type read by the base parser, without necessarily reading the sub-elements or text of that element. The elements of the XML document are then stepped through by the base parser in combination with the iterative method until the element to be extracted is located, read, and processed by the base parser. An event is generated, that is related to the element, and placed on an event stream for use by an application such as a Java application." (Paragraph [0031], which is cited by the Final Action).

3. The Appellants' Position regarding Independent Claims 1, 13, 25, and 37, and the Dependent Claims 38-53

Fry merely discloses a system and method for XML parsing, in which a name of an element type is extracted from an XML document. An element type tag of the XML document is then located and the element type read by the base parser, without necessarily reading the sub-elements or text of that element. The elements of the XML document are then stepped through by the base parser in combination with the iterative method until the element to be extracted is located, read, and processed by the base parser.

The present invention describes a system and method of query processing, in which tag identifiers are scanned to determine whether a match exists between any of the tag identifiers and the query, parsing a matched textual element only if the tag identifier matches the query, and skipping parsing of the unmatched textual element, when the tag identifier does not match the query.

In contrast, Fry does not disclose, teach or suggest query processing, as does the

present invention. Nowhere does Fry disclose, teach or suggest matching a query to a tag identifier of a mark-up language data stream, as does the present invention. Instead, Fry discloses locating and reading an element type tag of an XML document and then parsing the element by the base parser.

In addition, the present invention describes at least the feature of skipping the parsing of an unmatched textual element, if a tag identifier, corresponding to the unmatched textual element, does not match the query.

In contrast Fry discloses "[a]n element tag of the XML document is located and the element type read by the base parser, without necessarily reading the sub-elements or text of that element." (Paragraph [0031]). (emphasis added). That is, Fry always parses the extracted, located, and read element type, but he may not read the sub-elements.

Upon not matching the tag identifier, which might include an element type tag, to the query, however, the present invention skips parsing the unmatched textual element. After all, as is known to those in the art, matching is not parsing. Skipping the parsing relates to the present invention because "there is a need for a novel technique that can reduce parsing time in the context of processing XQuery queries over XML documents" (Specification, page 3, lines 1 and 2).

For at least the reasons outlined above, Applicants respectfully submit that Fry does not disclose, teach or suggest the present invention's features of: "A method for query processing ... scanning, by a processor, all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; parsing a matched textual element, if a tag identifier, corresponding to said matched textual element, matches said query; and skipping an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query", as recited in independent claim 1, "A system for query processing ... a processor adapted to scan all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; and a parser adapted to ... skip an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query", as recited in independent claims

13 and 37, and "A program storage device readable by computer comprising a program of instructions executable by said computer to perform a method for query processing ... scanning, by a processor, all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; parsing a matched textual element, if a tag identifier, corresponding to said matched textual element, matches said query; and skipping an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query", as recited in independent claim 25. Accordingly, Fry does not anticipate, nor render obvious, the subject matter of independent claims 1, 13, 25, and 37, and dependent claims 38-53 under 35 U.S.C. §102(e).

In view of the foregoing, the Board is respectfully requested to reconsider and withdraw the rejection of claims 1, 13, 25, and 37-53 under 35 U.S.C. §102(e) as anticipated by Fry.

VIII. CONCLUSION

In view of the foregoing, the Appellants respectfully submit that the cited prior art does not disclose, teach or suggest the present invention's features described by independent claims 1, 13, 25, and 37, and as such, claims 1, 13, 25, and 37 are patentable over Fry. Further, dependent claims 38-53 are similarly patentable over Fry, not only by virtue of their dependency from patentable independent claims, respectively, but also by virtue of the additional features of the Appellants' claimed invention they define. Thus, the Appellants respectfully request that the Board reconsider and withdraw the rejections of claims 1, 13, 25, and 37-53 and pass these claims to issue.

Please charge any deficiencies and credit any overpayments to Attorney's Deposit Account Number 09-0441.

Respectfully submitted,

Date: June 17, 2008

/Peter A. Balnave/
Peter A. Balnave, Ph. D.
Registration No. 46,199

Gibb & Rahman, LLC
2568-A Riva Road, Suite 304
Annapolis, MD, 21401
Voice: (410) 573-5255
Fax: (301) 261-8825
Balnave@Gibb-Rahman.com
Customer No. 29154

IX. CLAIMS APPENDIX

1. A method for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document, said method comprising:

producing, by said streaming API for a mark-up language data stream, an ordered index of all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements;

scanning, by a processor, all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers;

parsing a matched textual element, if a tag identifier, corresponding to said matched textual element, matches said query; and

skipping an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

2-12. (Canceled).

13. A system for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document, said system comprising:

an ordered index of all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming API for a mark-up language data stream;

a processor adapted to scan all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; and

a parser adapted to parse a matched textual element, if a tag identifier

corresponding to said matched textual element, matches said query, and to skip an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

14-24. (Canceled).

25. A program storage device readable by computer comprising a program of instructions executable by said computer to perform a method for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document, said method comprising:

producing, in said streaming API for a mark-up language data stream, an ordered index of all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements;

scanning, by a processor, all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers;

parsing a matched textual element, if a tag identifier, corresponding to said matched textual element, matches said query; and

skipping an unmatched textual element for parsing, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

26-36. (Canceled).

37. A system for query processing by using a streaming application programming interface (API) for a mark-up language data stream of a textual document, said system comprising:

an ordered index of all textual elements corresponding to their order in said mark-up language data stream, said ordered index comprising tag identifiers and end positions corresponding to each of said all textual elements and being produced by said streaming

API for a mark-up language data stream;

a processor adapted to scan all tag identifiers of said ordered index to determine if there exists a match between a query and any of said tag identifiers; and

a parser adapted to skip an unmatched textual element, if a tag identifier, corresponding to said unmatched textual element, does not match said query.

38. The method of claim 1, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprised a tag identifier and an end position.

39. The method of claim 1, all the limitations of which are incorporated herein by reference, further comprising storing a parsed matched textual element in a buffer.

40. The method of claim 1, all the limitations of which are incorporated herein by reference, wherein upon said skipping of said unmatched textual element for parsing, said method further comprises offsetting said parser based upon an end position stored in said ordered index and corresponding to said unmatched textual element.

41. The method of claim 1, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).

42. The system of claim 13, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprising a tag identifier and an end position.

43. The system of claim 13, all the limitations of which are incorporated herein by reference, further comprising a buffer, operatively connected to said parser, in which a parsed matched textual element is stored.

44. The system of claim 13, all the limitations of which are incorporated herein by reference, wherein said parser is offset based upon an end position stored in said ordered index that corresponds to said unmatched textual element.

45. The system of claim 13, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).

46. The program storage device of claim 25, all the limitations of which are incorporated herein by reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprises a tag identifier and an end position.

47. The program storage device of claim 25, all the limitations of which are incorporated herein by reference, further comprising storing a parsed matched textual element in a buffer.

48. The program storage device of claim 25, all the limitations of which are incorporated herein by reference, wherein upon said skipping if said unmatched textual element for parsing, said method further comprises offsetting said parser based upon an end position stored in said ordered index and corresponding to said unmatched textual element.

49. The program storage device of claim 25, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).

50. The system of claim 37, all the limitations of which are incorporated herein by

reference, wherein each of said textual elements comprises textual sub-elements, and each of said textual sub-elements comprising a tag identifier and an end position.

51. The system of claim 37, all the limitations of which are incorporated herein by reference, further comprising a buffer, operatively connected to said parser, in which a parsed matched textual element is stored.

52. The system of claim 37, all the limitations of which are incorporated herein by reference, wherein said parser is offset based upon an end position stored in said ordered index that corresponds to said unmatched textual element.

53. The system of claim 37, all the limitations of which are incorporated herein by reference, wherein said mark-up language data stream comprises HyperText Markup Language (HTML) or Extensible Markup Language (XML).

X. EVIDENCE APPENDIX

There is no other evidence known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There is no other related proceedings known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.